Appendix A: Custom Server-Side Code and SQL Date Formatting

Custom Server-Side Code and SQL Date Formatting

This section applies to developers writing **custom server-side Methods** that construct SQL queries directly, for example, Methods that call applySQL() or build SQL WHERE clauses containing date values as string literals. It does not apply to standard IOM API usage via setProperty() and apply().

> **Important**
>
> This advisory applies to SaaS deployments running on Aras Innovator Release 40 (.NET 10, Ubuntu 24.04). On-premises deployments are not affected unless the server runs on Ubuntu 24.04 or another platform using ICU 74.

Background

As noted in Cross-Platform Development, date and time formatting in .NET behaves differently depending on the operating system. Aras Innovator Release 40 runs on .NET 10 hosted on Ubuntu 24.04. The ICU 74 globalization library shipped with Ubuntu 24.04 introduced a change in certain locales where a Narrow No-Break Space (U+202F) is used instead of a standard ASCII space before the AM/PM designator in formatted date strings.

When a server-side Method passes a locale-formatted date string directly to SQL Server as part of a query condition, rather than using a format-neutral string or a typed parameter, SQL Server cannot parse it and throws an exception.

Who Is Affected

Developers with custom server-side Methods that:

- Pass a locale-sensitive date string directly to SQL Server as part of a SQL condition
- Call applySQL() with date values produced by JavaScript's new Date().toString() or .NET's default DateTime.ToString() without an explicit format
- Construct SQL WHERE clauses with date string literals instead of parameterized queries

No action is required for:

- Standard Aras Innovator product usage
- Custom Methods that use setProperty() with dates in yyyy-MM-ddThh:mm:ss format (see How to Handle Date Properties)
- On-premises deployments not running Ubuntu 24.04

Example of Affected Code

The following server-side JavaScript pattern is at risk:

```
// AVOID: locale-sensitive date passed directly to SQL
var myDate = new Date();
var dateStr = myDate.toString(); // Produces locale-specific string, may include U+202F

var result = this.getInnovator().applySQL(
    "SELECT id FROM innovator.[Part] WHERE created_on > '" + dateStr + "'"
);
```

The Date.toString() output varies by locale and platform. On Ubuntu 24.04 with ICU 74, certain locales produce a Narrow No-Break Space (U+202F) before the AM/PM designator. SQL Server cannot parse this and throws a conversion exception.

What To Do

Use the yyyy-MM-ddThh:mm:ss format explicitly at all SQL boundaries. This is the same format required by setProperty() throughout the IOM API (see How to Handle Date Properties):

```
// CORRECT: locale-neutral format
var myDate = new Date();

function dateFormat(d) {
    var s = d.getFullYear() + "-";
    s += pad(d.getMonth() + 1) + "-";
    s += pad(d.getDate()) + "T";
    s += pad(d.getHours()) + ":";
    s += pad(d.getMinutes()) + ":";
    s += pad(d.getSeconds());
    return s;
}

function pad(x) { return (x < 10) ? "0" + x : "" + x; }

var result = this.getInnovator().applySQL(
    "SELECT id FROM innovator.[Part] WHERE created_on > '" + dateFormat(myDate) + "'"
);
```

The dateFormat function above produces a culture-invariant string (2026-07-06T14:48:00) that SQL Server parses correctly on all platforms. Note that this is identical in structure to the dateFormat helper shown in How to Handle Date Properties for use with setProperty().

Where possible, prefer parameterized queries over string concatenation entirely, as they are safer against both this issue and SQL injection.

Relationship to Cross-Platform Development and How to Handle Date Properties

The section Cross-Platform Development documents the general cross-platform date formatting difference between Windows and Linux.
The section How to Handle Date Properties documents the correct yyyy-MM-ddThh:mm:ss format for setProperty().
This appendix addresses the compatibility issue that arises when custom code bypasses setProperty() and constructs SQL strings directly using unformatted date values.

Further Reference

- Aras Innovator 40: Upgrading to 40 from Aras Innovator 14+, Appendix A: Custom Server-Side Code and SQL Date Formatting
- Aras IOM SDK 15.1.2 Release Notes, Cross-Platform Date/Time Formatting Normalization